User Manual - Version 1.1

Results Through Focused Technology...

DataStrike! © *1992-1998 Soft Solutions, Inc. All Rights Reserved.*
Published World-Wide by Soft Solutions, Inc.

Soft Solutions, Inc.
2900 Chamblee Tucker Road
Building 12, Suite 200
Atlanta, GA 30341 USA
Web address: www.softsinc.com
Product sales e-mail: sales@softsinc.com
Product Support e-mail: support@softsinc.com

Written by: Bob Keleher
Concept & Design: Tim Haratine

Software License and Limited Warranty

DataStrike! Customers: IMPORTANT—READ CAREFULLY BEFORE USING.

By installing this software program, you indicate your acceptance of the following Soft Solutions, Inc. License Agreement.

Software License and Limited Warranty Agreement:

This is a legal agreement between you, the end user, and Soft Solutions, Inc. By licensing this package, you are agreeing to be bound to the terms of this agreement. If you do not agree to the to the terms of this agreement, promptly return all program materials (including written materials and binders or other containers) to the place you obtained them for a full refund.

Soft Solutions, Inc. Software License

1. GRANT OF LICENSE. Soft Solutions grants to you the right to use one development copy of the enclosed Soft Solutions software program (the "SOFTWARE") on a single terminal connected to a single computer (i.e. , with a single CPU), or on a LICENSED COMPUTER NETWORK. Each concurrent user of the SOFTWARE must have exclusive access to a Soft Solutions SOFTWARE manual during his/her use. Soft Solutions, Inc. as Licensor, grants to you, the LICENSEE, a non-exclusive, non-transferable right to use this Software subject to the terms of the license as described in the following sections:

A. You may make backup copies of the Software for your use provided they bear the Soft Solutions, Inc copyright notice.

B. You may use this software in an unlimited number of custom or 4D-compiled commercial database applications created by the original licensee. No additional product license or royalty is required. You are not permitted to sell any 4D source code containing any portion of DataStrike! to another party without first obtaining a Developer license agreement for the intended party.

2. COPYRIGHT, The SOFTWARE is owned by Soft Solutions or its suppliers and is protected by United States copyright laws and international treaty provisions. Therefore, you must treat the SOFTWARE like any other copyrighted material (e.g., a book or musical recording) except that you may either (a) make one copy of the SOFTWARE solely for backup or archival purposes, or (b) transfer the SOFTWARE to a single hard disk provided you keep the original solely for backup or archival purposes. You may not copy the written materials accompanying the SOFTWARE.

3. OTHER RESTRICTIONS. You may not rent or lease the SOFTWARE, but you may transfer the SOFTWARE and accompanying written materials on a permanent basis provided you retain no copies and the recipient agrees to the terms of this Agreement. You may not reverse engineer, decimalize, or decompile the DataStrike! plug-in SOFTWARE. If SOFTWARE is an update, any transfer must include the update and all prior versions.

LIMITED WARRANTY. Soft Solutions, Inc. warrants that (a) the SOFTWARE will perform substantially in accordance with the accompanying written materials for a period of ninety (90) days from the date of receipt. Any implied warranties on the SOFTWARE are limited to ninety (90) days and one (1) year, respectively. Some states do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.

CUSTOMER REMEDIES. Soft Solutions, Inc. entire liability and your exclusive remedy shall be, at Soft Solutions option, either (a) return of the price paid or (b) repair or replacement of the SOFTWARE that does not meet Soft Solutions Limited Warranty and which is returned to Soft Solutions with a copy of your receipt. This Limited Warranty is void if failure of the SOFTWARE has resulted from accident, abuse, or misapplication. Any replacement SOFTWARE will be warranted for the remainder of the original warranty period of thirty (30) days, whichever is longer.

NO OTHER WARRANTIES. Soft Solutions disclaims all other warranties, either express or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose, with respect to the SOFTWARE, the accompanying written materials, and any accompanying hardware. This limited warranty gives you specific legal rights. You may have others, which vary from state to state.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES. In no event shall Soft Solutions or it suppliers be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use of or inability to use this Soft Solutions product, even if Soft Solutions has been advised of the possibility of such damages. Because some states do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

Governing Law. This entire agreement shall be governed by the laws of the State of Georgia.

DataStrike! is a trademark of Soft Solutions, Inc.
Apple is a registered trademark of Apple Computer, Inc.
Windows is a registered trademark of Microsoft, Inc.
4th Dimension, 4D Compiler, 4D External Mover and 4D are registered trademarks of ACI and ACI US, Inc.

Table of Contents

Figure Reference vii

Introducing DataStrike!™ 1

Installing DataStrike! 2

Integrating DataStrike! into Your Database 2

Coding Requirements 2

Mandatory Developer Steps 3

1. Move DataStrike! Libraries with 4D Insider 3

2. Setup and Initialization 3

Manual or Automatic Relations 4

3. Product Registration 4

4. Providing User Access to the DataStrike! Editor 4

Optional Developer Steps 5

1. Updating the DataStrike! table map when structural changes occur 5

2. DataStrike! Query Editor pop-up list support for database fields 6

DataStrike! Localization 7

4D's Layout Editor - (What you will see) 12

Using DataStrike! 14

The DataStrike! Query Editor 14

Loading, Editing and Saving Query Filters 15

DataStrike! Query Editor Components 15

The Table Pull-down Menu 15

Fields, Comparison Operators and Values 16

The Functions Button 18

Query Argument Editing 20

Grouping 20

Editing Query Arguments 21

Conjunction Operators (And, Or and Except) 22

The And Conjunction 23

The Or Conjunction 23

The Except Conjunction 24

Set Operators 24

Executing Queries and Query Filters 25

Circular Relationships 28

Solving Complex Circular Path Problems 30

Severing a "Many" table link 31

Severing a Table 31
Index.....33

Figure References

Figure 1: DataStrike! Insider Library and 4D Insider 6.0.5 3
Figure 2: DataStrike! Insider Library 3
Figure 3: Query Editor Pop-up lists 6
Figure 4: Language Localization 7
Figure 5: DataStrike! STR# Resources 7
Figure 6: 27501 - DataStrike! Query Editor Resource Strings 8
Figure 7: 27502 - Resource Strings used Within Other DataStrike! Forms 8
Figure 8: 27503 - Method Resource Strings 9
Figure 9: Form Design - Show Format 12
Figure 10: Form Design - Show Resource 12
Figure 11: Form Design - Show Name 13
Figure 12: The DataStrike! Query Editor 14
Figure 13: The DataStrike! Query Filters Dialog 15
Figure 14: Table Pulldown Menu 15
Figure 15: Field Selection Scrollable Area 16
Figure 16: Field Pop-up Choice List 17
Figure 17: Comparison Operators 17
Figure 18: Possible Uses of Function Option 18
Figure 19: Function Dialog 19
Figure 20: Argument Editing Buttons 20
Figure 21: Create Group Buttons 20
Figure 22: Query Example #1 21
Figure 23: Query Example #2 21
Figure 24: Query Example #3 21
Figure 25: Clear All Button 21
Figure 26: Delete Line Button 22
Figure 27: Insert Line Button 22
Figure 28: Add Line Button 22
Figure 29: Conjunction Operators 22
Figure 30: Set Operators 24
Figure 31: The Query Button 25
Figure 32: The Query Filters Dialog 26
Figure 33: The Circular Path Dialog 26
Figure 34: Electing to save a Specific Path within a Query Filter 27
Figure 35: 2 Circular Paths 28
Figure 36: 3 Circular Paths 29
Figure 37: Multiple Non-adjacent Circular paths 29
Figure 38: Multiple Adjacent Circular paths 29

Figure 39: Circular paths with Subtables.....30
Figure 40: Complex Circular Paths (which are not allowed) 30

Introducing DataStrike!™

DataStrike! is a 4th Dimension v6 cross platform “source code” solution which replaces the 4D Query Editor. Fully relational, DataStrike! significantly enhances the end-user experience and value of any 4D Database by providing complete and optimized access to any related tables. By adding DataStrike! to a 4D database, Users now have a whole new horizon of questions they can ask. DataStrike! will dramatically increase the intrinsic and informational value of your 4D databases.

With DataStrike!, dead-end queries and labor-intensive coding tasks become part of the past.

Here’s why:

- Because DataStrike! is generic, it can easily integrate into any 4D database.
- Because DataStrike! queries can be saved at the table level, important queries are always a click away.
- Because DataStrike! “Saved” queries are records, developers can conveniently package valuable saved queries to customers.
- Because DataStrike! is optimized for the relational data model, 4D queries execute in the shortest possible time.[1]
- Because DataStrike! is steeped in logic, multiple argument line queries can be grouped, ordered and organized so all legitimate end-user questions can be answered.
- Because DataStrike! has built-in “Set” Management and support for up to 255 arguments per query, searching is now a one-stop proposition.
- Because DataStrike! has a clean, intuitive interface - end-users can get down to work immediately while increasing their productivity and improving their business.
- Because DataStrike! supports circular or “multipath” relationships, your users can always be sure they are getting the correct answer while knowing the precise question they are asking.
- Because DataStrike! gives users access to any of their data, your development effort for creating custom queries is finished once DataStrike! is installed.

[1]DataStrike! does not currently track statistical information on table sizes, indices, etc. A query could likely execute faster if database and table statistics were available.

Installation

Installing DataStrike!

This section outlines the requirements for copying and integrating the DataStrike! Insider Library into an existing 4th Dimension database.

Integrating DataStrike! into Your Database

The entire DataStrike! Query Engine library consists of 1 4D Table, 1 4D cross-platform plug-in (with 5 commands), nearly four dozen 4D methods and roughly one dozen 4D forms. Prior to moving the DataStrike! components into an existing 4D database with 4D Insider, you should ensure there are no naming conflicts between DataStrike! and the target 4D database. Once you have verified no naming conflicts exist (or remedied all conflicts), all DataStrike! objects can be moved into the database via 4D Insider.

All DataStrike! Methods and variables are prefixed with the characters “QE”.

There are 5 (five) 4D methods which have particular importance to developers using DataStrike!. These methods are shown in the following table:

Method Name	Description
QE ADD CHOICE LIST	Call this method to assign 4D lists to fields in DataStrike!
QE ON STARTUP	Initializes DataStrike! plug-in and ◊QEiDataBuildNo interprocess variable. Important! Call from “On Startup” and from “On Server Startup” 4D methods.
QE PREFERENCES	Establishes preferences for: “Circular Path” defaults, “field list sort order” defaults, Language Resource locations, “Show Filter Information” defaults and link exclusion.
QE CALL FROM FORM	Lets user run/edit saved query filters or create new queries.
QE UPDATE DATA	Forces rebuild of QE_Data table & associated arrays.

Coding Requirements

Installing DataStrike! is easy. Coding is not required, however, there are specific requirements related to the proper placement of DataStrike! methods and buttons in your 4D Application.

Four developer steps for completing DataStrike! integration are mandatory and several are optional. These requirements include moving DataStrike! into your database structure via 4D Insider and then some “copy and paste” steps inside your 4D database. Please take time to

properly implement the steps in the “Mandatory Developer Steps” section. DataStrike! is dependent on these.

The “Optional Developer Steps” are also important and may become necessary as your database evolves or if you want to customize your DataStrike! implementation. Please take a minute to read these sections and learn the steps necessary for your implementation.

Mandatory Developer Steps

1. Move DataStrike! Libraries with 4D Insider

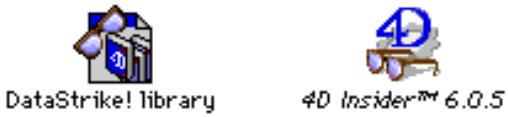


Figure 1: DataStrike! Insider Library and 4D Insider 6.0.5

The DataStrike! Library was built using 4D Insider v6.0.5 for Macintosh. Your first step is to load the DataStrike! Insider library and your target 4D database with 4D Insider. Once both are loaded and you have verified no naming or resource conflicts exist, it will be safe to move the DataStrike! library components into your 4D application.

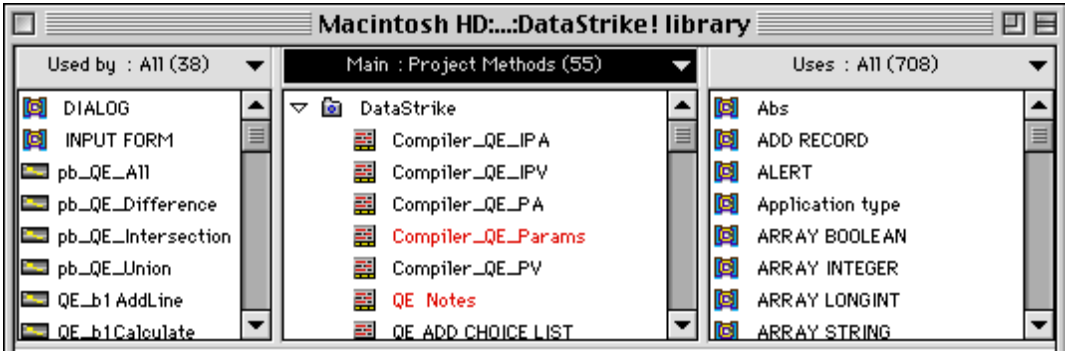


Figure 2: DataStrike! Insider Library

Please consult your 4D Insider manual for instructions on copying the DataStrike! libraries into your 4D application(s).

2. Setup and Initialization

The QE ON STARTUP method must be added into your database “On Start-up” and “On Server Startup” methods. “QE ON STARTUP” takes care of all DataStrike! dependent setup and initializations.

Similar to 4D's Editor's, DataStrike! only displays visible tables and fields. The QE ON STARTUP method initiates the 4D methods which map all visible table and field information - including subtables and their fields (provided they are visible).

Table names and field names containing an underscore, "_" are replaced with a blank character in the DataStrike! Query Editor.

- Manual or Automatic Relations

DataStrike! requires either a manual or automatic relation (or link) between two tables before considering them "related". Tables which are "related" are considered within the same path to one another. When the DataStrike! Editor is loaded, the available table list is loaded. This includes the current table along with all other related tables.

3. Product Registration

The small cross-platform DataStrike! plug-in provided with DataStrike! is mandatory as it requires special commands to build the database maps, manage subtables and enable developer registration. Ensure you place the DataStrike plug-in within the appropriate "4DX" folder(s).

The QE_Register command requires three (3) parameters as follows:

$String:=QE_Register("Developer";"Mac reg#";"Windows reg#")

Failure to register DataStrike! at program start-up will disable the remaining DataStrike! plug-in commands and render the DataStrike! Editor unusable.

4. Providing User Access to the DataStrike! Editor

To invoke DataStrike! capabilities from any output form, copy the button found in the "[QE_Data]Output Button" form to your output form(s). This button calls the Object method "QE_bCallQE". Once this button object is placed on your output form, the end-user can simply click on the button to access the DataStrike! Query Editor.

Optional Developer Steps

1. Updating the DataStrike! table map when structural changes occur

The QE_Data table is where all required database relationships are identified for the proper functioning of the DataStrike! Query Editor. Should any database (which uses DataStrike!) structurally change, the QE_Data table records must also be updated. Updating the QE_Data table is accomplished by calling the “QE UPDATE DATA” method from a menu item, an object or any other valid approach to executing a 4D method. For vertical market applications, for example, the “QE UPDATE DATA” method might be called procedurally.

A database structure change could be a field name change, a new data relationship or a new table. The QEs_Rebuild_Check method will check to ensure existing links and the current number of tables are preserved from the current settings. Developers should run QE UPDATE DATA whenever the structure has been changed.

The QE_Data table contains three (3) different type of records.

1. The first record type consists of “mapping records”. Mapping records are identified by negative table number fields in the QE_Data table. They contain structure mapping and optimal “navigation” information required for traversing a relational database. Specifically, “mapping” records provide the table and field information necessary to perform Relate One Selection commands (formerly “Join”) and the Relate Many Selection commands (formerly “Project Selection”).
2. The second type of record is the “version” record. There is only one (1) version record per database and it has a table number of zero (0) in the table number field. Built from 4D methods, the version number helps Soft Solutions understand which version of the program you are using.
3. The third record type is for storing Query Filters. Each Query Filter record is identifiable by a positive table number in the table number field. The Query Filter record type contains the query argument information along with the set, selection and path information critical to proper query building and execution. Query Filters are created and saved by database end-users.

Whenever the QE UPDATE DATA method is executed, existing Query Filters are preserved in the QE_Data table. All mapping records, however, are deleted and rebuilt to fully identify the new structure.

2. DataStrike! Query Editor pop-up list support for database fields

DataStrike! provides developer assignable choice lists for any database field.

Figure 3: Query Editor Pop-up lists

To assign a list to any field, call the “QE_AssignList” method (for each field) as follows:

QE ADD CHOICE LIST(->[Table1]Fieldname1;"Listname1”)
QE ADD CHOICE LIST(->[Table2]Fieldname2;"Listname2”)

These calls should be made at 4D Start-up and prior to the loading of the DataStrike! Query Editor, but after the startup method has been completed.

To remove a list assigned to a field, call the “QE_AssignList” method (for each field) as follows:

QE ADD CHOICE LIST(->[Table1]Fieldname1;"")

Because the QE ADD CHOICE LIST method adds the list information directly to the QE_Data table, it is only necessary to assign lists to a field one time.

DataStrike! Localization

For the purpose of customizing or localizing, DataStrike!-specific application strings contain string values which are retrieved from the “STR#” resource type within your 4D database Structure. Within the STR# resource type, DataStrike! uses resource ID numbers 27501, 27502 and 27503. These resource ID #’s are specific to DataStrike! and reference to them is made in the 4D Method named “QE PREFERENCES”.

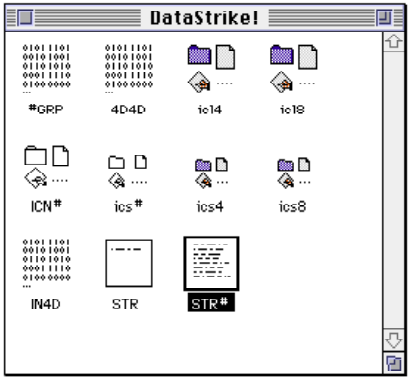


Figure 4: Language Localization

If you must change any of the text strings or localize them to a language other than English, make the appropriate modifications with “ResEdit” to the 27501-27503 STR# resources in your 4D application. If these string resources are in conflict with a resource numbering system currently in use, it will be necessary for you to copy the 27501-27503 resource numbers into other numbers within the STR# resource. Any resource numbering changes must then also be made to the QE PREFERENCES method in your 4D application (please see code snippet below). Upon request, Soft Solutions can provide a complete list of the strings contained within the 27501 - 27503 resource types.

```
◊QEkl_EditorForm:=27501  `text for Query Editor form
◊QEkl_OtherForms:=27502  `text for other forms
◊QEkl_MethodStrings:=27503  `text for method strings
```

Figure 5 below shows the Resource ID #’s the 4D developer will see within Resedit.

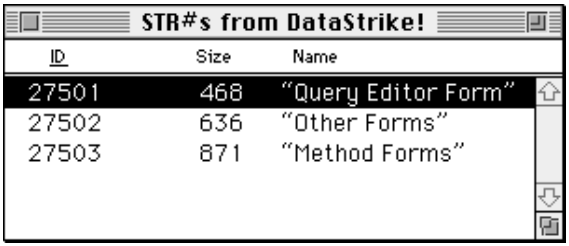


Figure 5: DataStrike! STR# Resources

Double-click in the 27501 Resource ID# to look at the resource strings used within the DataStrike! Query Editor.

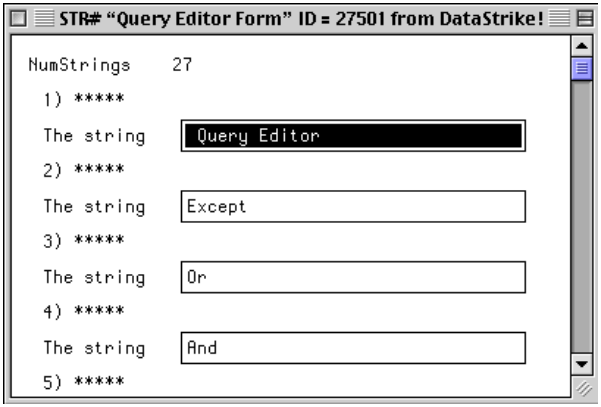


Figure 6: 27501 - DataStrike! Query Editor Resource Strings

Now double-click in the 27502 Resource ID# to look at the resource strings used within the associated forms and dialogs used by the DataStrike! Query engine (see Figure 7 below).

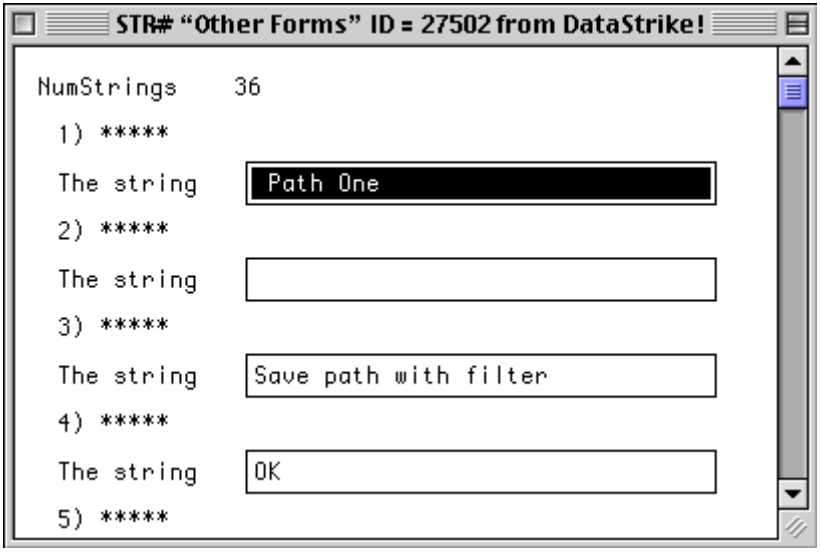


Figure 7: 27502 - Resource Strings used Within Other DataStrike! Forms

Last, double-click in the 27503 Resource ID# to look at the resource strings used within DataStrike! Query engine methods (see Figure 8 below). Unlike Resource #'s 27501 and 27502, the 27503 resources are not obvious and require some hunting around in various DataStrike! methods before learning the details used to build these strings. For this reason, a detailed table outlining the various strings has been provided following Figure 8.

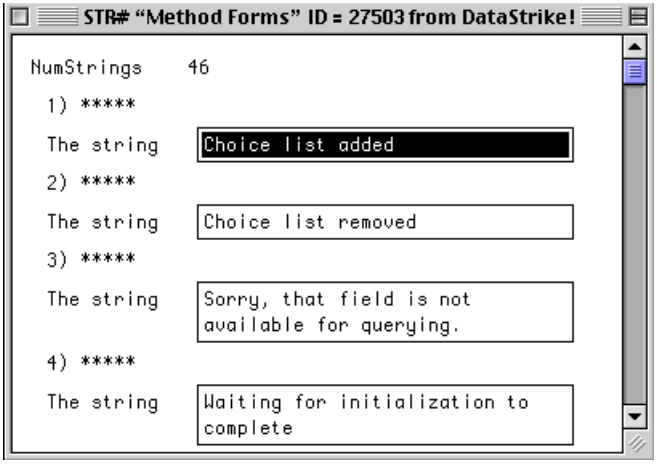


Figure 8: 27503 - Method Resource Strings

Please review the following table for specific Strings contained within the STR# resource type for ID# 27503. These strings are contained within various DataStrike! methods and forms. Their location (method or form), string number, string value and comments are shown in the table. Where "form" references are indicated in column 1, the string is used in an object on the form.

STR# Resource Table (27503) for DataStrike! Methods

Method where first used	27503 String#	String	Comments
QE ADD CHOICE LIST	1	Choice list added	
	2	Choice list removed	
	3	Sorry, that field is not available for querying.	
QE CALL FROM FORM	4	Waiting for initialization to complete.	
	5	Updating Query Engine information.	
	6	Unable to update the server. Try restarting the server.	
	7	Query Filters	Dialog title

Localization

Method where first used	27503 String#	String	Comments
	8	Sorry, that filter has been deleted.	
	9	AND	
	10	OR	
	11	EXCEPT	
	12	and	
	13	or	
	14	except	
	15	FIELD NOT AVAILABLE!	
	16	Count	
QEe_HandleFx	17	Sorry, you can only perform calculations on related many tables	
	18	This is not a related table.	
	19	This is not a many table.	
	20	Please select a line to edit	
	21	Function	
QE_HandleFXInCircle	22	to	
QEe_QueryEditor	23	Query	
QEs_Build_Comparisons	24	is equal to	Max 27 chars
	25	is not equal to	Max 27 chars
	26	is greater than	Max 27 chars
	27	is greater than or equal to	Max 27 chars
	28	is less than	Max 27 chars

Method where first used	27503 String#	String	Comments
	29	is less than or equal to	Max 27 chars
	30	begins with	Max 27 chars
	31	contains	Max 27 chars
	32	does not contain	Max 27 chars
	33	select a field	Max 27 chars
	34	Waiting for a filter record	
	35	Save path with filter	
	36	Untitled filter	
	36	Untitled filter	
	37	Please select a field first	
	38	My filter	default new filter name when saving
QE_LockDataRecord	39	Save filter	
Form: CircularPath	40	Replace existing	
Form: FilterLoad	41	Query filters	
Form: QueryEditor	42	Sorry, that filter has been deleted	
QEe_BuildLine	43	And	All same length, pad with spaces (these are all 5 characters long)
	44	Or	All same length, pad with spaces (these are all 5 characters long)
	45	Exc	All same length, pad with spaces (these are all 5 characters long)
QEe_HandleFxInCircle	46	Circular Path	

4D's Layout Editor - (What you will see)

The Show Format menuitem in Figure 9 below displays the [QE_Data]QueryEditor form variables displaying the actual resource name and number.

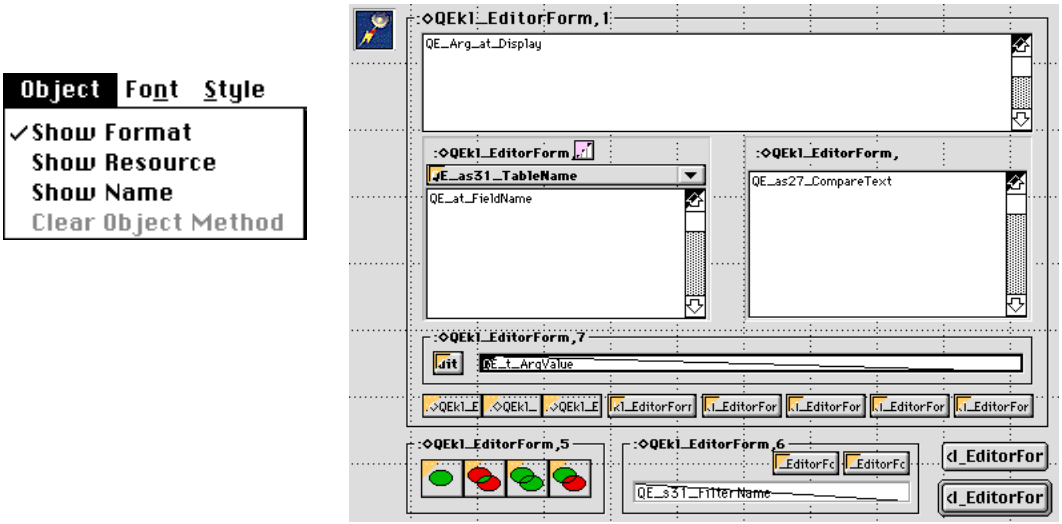


Figure 9: Form Design - Show Format

The Show Resources menuitem in Figure 10 below displays the [QE_Data]QueryEditor form variables after reconciling the actual resource name. If you are translating or customizing resources, this is how you would audit any changes to ensure they display correctly.

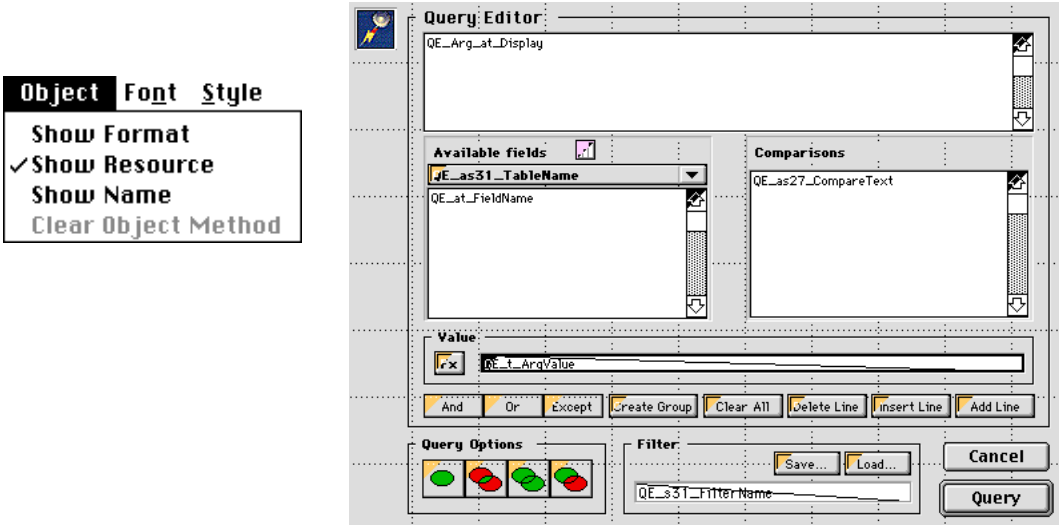


Figure 10: Form Design - Show Resource

The Show Name menuitem in Figure 11 displays [QE_Data]QueryEditor form variables with the actual resource name. For example, ◊QEkl_EditorForm, which is assigned the Resource ID of 27501 within QE PREFERENCES, is followed by a “, 1”. In other words, “27501,1”. This tells 4D to look in STR# 27501 and display item number 1. STR# 27501 used is for the Query Editor.

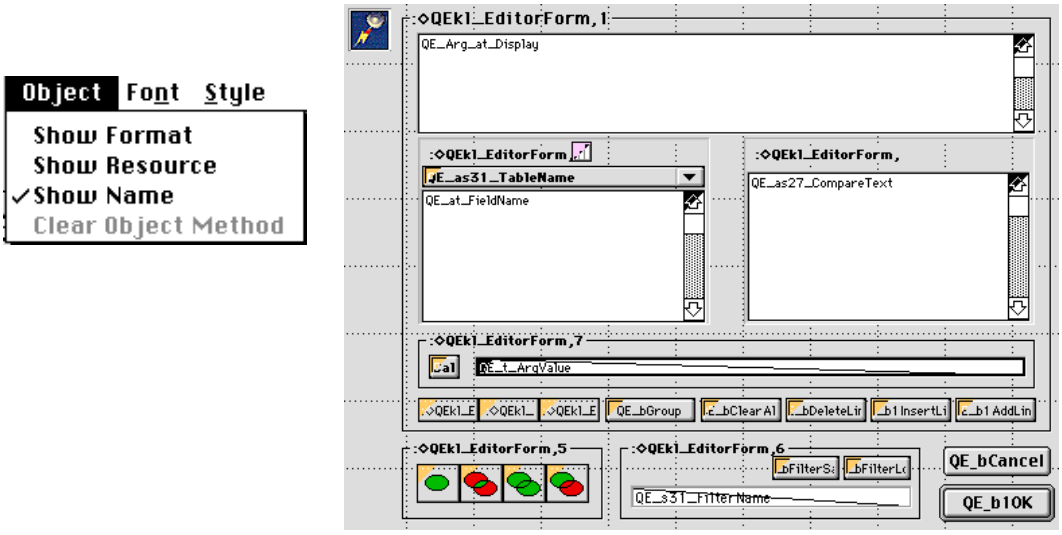


Figure 11: Form Design - Show Name

Using DataStrike!

The DataStrike! Query Editor

The End-User is presented with the DataStrike! Query Editor (shown in Figure 12 below) when no Query Filters exist or when they select a filter and click the Edit button.

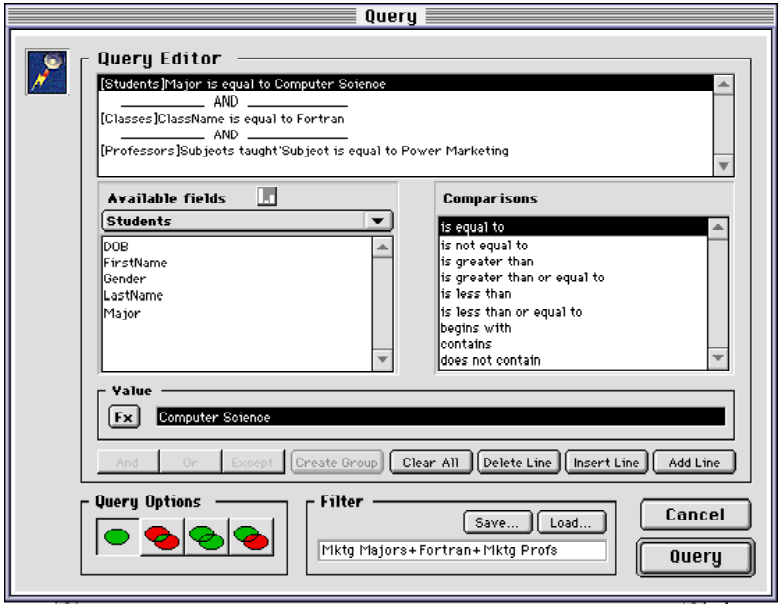


Figure 12: The DataStrike! Query Editor

The DataStrike! Query Editor has a simple appearance similar to the 4th Dimension version 6 Query Editor. Beyond the similarity in appearance, the overall query capabilities are distinct.

Developers or End-users can copy the current Query (all argument lines) to the Clipboard by depressing Control-C on Windows machines or Command-C on the Macintosh.

Values displayed in the query lines are immediately updated as values are typed in. This process is automatic and takes advantage of new commands in 4th Dimension version 6.

Loading, Editing and Saving Query Filters

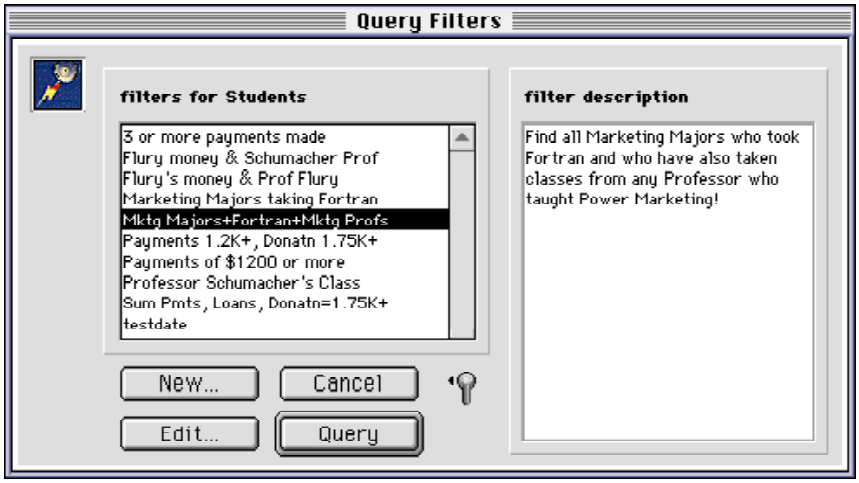


Figure 13: The DataStrike! Query Filters Dialog

To edit an existing “filter” within the DataStrike! Query Editor, click on the “Load” button and select the filter you want loaded. The filter will then be loaded and the Query can be “Run”. Alternatively, the same Query filter can be edited and saved - overwriting the previous version. Alternatively, it can be saved as a new Query filter by altering the filter name in the filter name dialog. There is no practical limit to the number of Query filters a 4D table can have.

DataStrike! Query Editor Components

- **The Table Pull-down Menu**

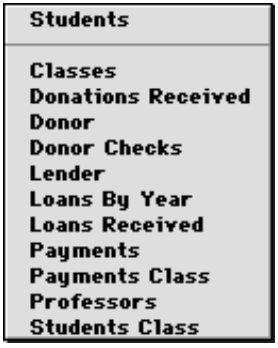


Figure 14: Table Pulldown Menu

The DataStrike! table pull-down menu is WYSIWIG. In other words, unlike the 4D Query Editor, if you can see a table from the DataStrike! table pull-down menu, you can query against

it. The table pull-down is formatted so the current table is always listed first - above the separator line. All other related tables (either directly or indirectly) are shown below the separator line in alphabetical order.

■ **Fields, Comparison Operators and Values**

For every Query, at least one Query Argument must be specified. A Query Argument consists of the field you wish to search on, the comparison operator and the Query Value. The comparison operator tells 4D how the value will be compared to the field contents.

Field Selection

The Field List initially reflects the fields for the table shown in the pulldown menu. Fields are initially shown in their physical (database) order unless specified differently in the “QE PREFERENCES” method. Clicking on the chart-like icon will reorder the fields in alphabetic or physical order. Note the Icon/button behaves like a toggle and flips the sort order between the two options.

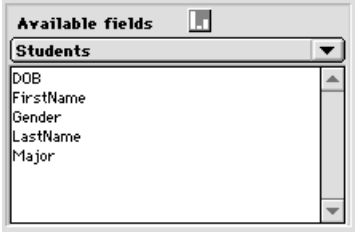


Figure 15: Field Selection Scrollable Area

The first thing you will want to do when creating or building a query is to select a field. The field list initially shown (see Figure 15) contains fields for the current table. Alternatively, you can switch tables by selecting the Table pulldown menu and selecting another table. After another table is selected, the field list for the selected table will be redrawn in physical order.

Choice Lists

Pop-up lists appear automatically whenever a User selects a field which has been assigned a pop-up list (pop-up lists are Developer assigned and reflect an existing 4D list assigned to any visible field). Sorted lists support User type-ahead for fast choice selection. The pop-up dialog provided within DataStrike! allows user-directed additions and changes.

Figure 16: Field Pop-up Choice List

DataStrike! also supports lists assigned to Boolean fields. For example, in lieu of True or False, the User can create and assign a list to a “Gender” field containing two values of “Male” or “Female”. Take a look at page 2 of the [QE_Data]QueryEditor to see how this is done. With Boolean fields, it is important to only assign two choices to a list.

Comparison Operators

The “Comparison Operators” area is dynamically built to reflect a valid list of operators based on the field “type” selected. For instance, the “Contains” Comparison Operator is only valid for string or text fields and would not be shown for numeric, boolean, date or time fields.

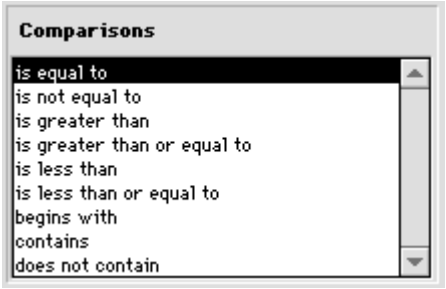


Figure 17: Comparison Operators

"Comparison Operators" tell the DataStrike! Query engine how to query the selected field given the value entered.

Values

"Values" refers to a string, number, date or time value you want to query on from the database table(s) specified.

■ The Functions Button

The "Fx" button stands for "Function" and allows a 4D user to perform calculations on a series (or set) of values from related 4D records.

The Function option is only available for use on tables which are "Many" tables with respect to its immediate neighbor table (in the path towards the target table). This requirement ensures record "grouping" will occur on the related "Many" table. Once the related "Many" selection is established, the Function logic derives the corresponding records for the related "One" table. Look at your possible options in Figure 18.

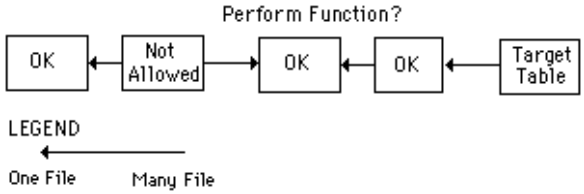


Figure 18: Possible Uses of Function Option

The Function button can be used for locating sets of related 4D "Many" table records meeting criteria such as Sum, Average, Min or Max, a numeric field must be chosen. Clicking the Function button displays the Function dialog shown on the following page.

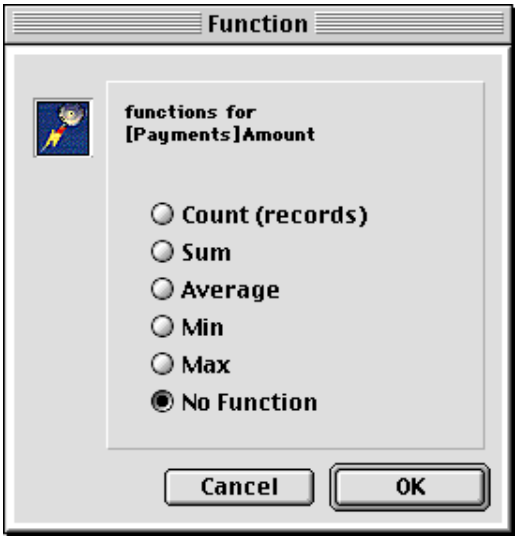


Figure 19: Function Dialog

Sum: Used to locate a series of related records whose numeric field "Sum" reflects the comparison operator and value entered.

Example: Sum([Invoices]BalanceDue>5000) 'Have Account Rep contact them asap!

Average: Used to locate a series of related records whose specified numeric fields have an "Average" reflective of the comparison operator and value entered.

Example: Average([Invoices]DaysLate>30) 'Put them on incentive program

Min: Used to locate a series of related records whose smallest numeric field minimum reflects the comparison operator and value entered.

Example: Min([Invoices]CurrentBalance=1) 'Holdover billing until next month

Max: Used to locate a series of related records whose numeric field "Sum" reflects the comparison operator and value entered.

Example: Max([Invoices]OrderTotal>10000) 'Place them on preferred customer list

When selecting the **Count** function, however, any field can be chosen because Count is performed at the record level and not at the field level. The “Count” Function counts records whose record count matches the search criteria entered.

For example:

In a hypothetical database where a Companies table is a “One” table and an the Invoice table is a “Many” table to Companies, a 4D User performs the following Query on/from the Companies table.

“Count [Invoice]Balance is greater than or equal to 3"

This query retrieves all Company records which have 3 or more related Invoice records. Expanding upon this query, an additional argument containing a numeric Function is added.

“Sum([Invoice]Balance)>5000 And
Count [Invoice]Balance is greater than or equal to 3"

The above query retrieves all Company records whose [Invoice]Balances are greater than $5000 and which have 3 or more related Invoice records.

Query Argument Editing

Figure 20: Argument Editing Buttons

■ **Grouping**

An essential aspect of DataStrike! that provides the power to query across many different tables within the same “Query” is the “Create Group” feature. DataStrike! automatically creates groups whenever a new 4D Table is introduced within a subsequent Query argument line. However, DataStrike! does not automatically group consecutive Query argument lines belonging to the same 4D table. The 4D User is permitted to segregate between consecutive same-table query argument lines by clicking on the “Create Group” button.

Figure 21: Create Group Buttons

Once the groups are established, the queries can be performed individually and appropriate “Set” logic can be applied to the resulting record sets.

These combinations any combination of different (related) tables, conjunctive logic and set manipulation providing a full breadth of query options.

For example, below is a 3 argument Query where all query arguments are for the “Students” table.

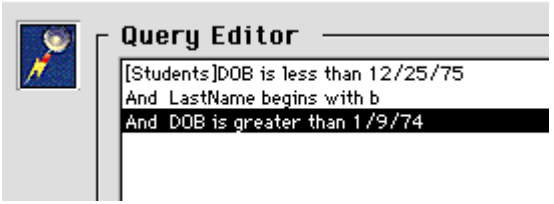


Figure 22: Query Example #1

Notice, through the use of the “Create Group” button, how an End-User can change the question (query) by grouping the three (3) arguments differently.

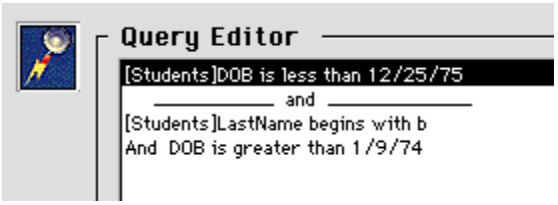


Figure 23: Query Example #2

Notice again, through the use of the “Create Group” button, how an End-User can change the question.

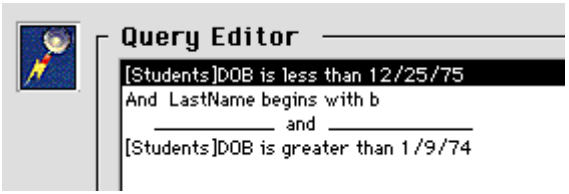


Figure 24: Query Example #3

■ Editing Query Arguments

Clear All

Click this button to clear all query arguments shown in the current query.

Figure 25: Clear All Button

Delete Line

Click this button to delete the currently selected (highlighted) argument line shown in the current query.

Figure 26: Delete Line Button

Insert Line

Click this button to insert a new line in front of the currently selected (highlighted) argument line.

Figure 27: Insert Line Button

Add Line

Click this button to add a new line after the currently selected (highlighted) argument line.

Figure 28: Add Line Button

Conjunction Operators (And, Or and Except)

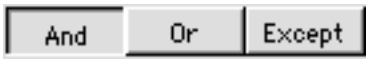
Figure 29: Conjunction Operators

To show how Conjunction operators are used, consider the following Customers database:

The following records comprise all records within the Customers table:

Lastname	Firstname
Glenn	Adams
Glenn	Smith
Steve	Adams
Steve	Smith

■ The And Conjunction



The “And” Conjunction Operator specifies a relationship between two Query arguments (or two Query Groups) where “Intersection” logic is used between the results of each of the two arguments. Here is an example of what happens:

In the following query:

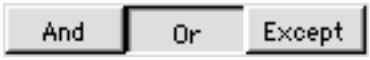
> Query [Customers]Lastname is equal to Smith and
> [Customers]Firstname is equal to Glenn.

The “And” operator tells DataStrike! to find those records only when both conditions are met.

The following records would be shown:

Lastname	Firstname
Glenn	Smith

■ The Or Conjunction



The “Or” Conjunction Operator specifies a relationship between two Query arguments (or two Query Groups) where “Union” logic is used between the results of each of the two arguments. Here is an example of what happens.

In the following query:

> Query [Customers]Lastname is equal to Smith Or
> [Customers]Firstname is equal to Glenn.

The “Or” operator tells DataStrike! to find those records where either condition is met.

Lastname	Firstname
Glenn	Adams
Glenn	Smith
Steve	Smith

- The Except Conjunction

The “Except” Conjunction Operator specifies a relationship between two Query arguments (or two Query Groups) where “Difference” logic is used between the results of each of the two arguments. Here is an example of what happens:

In the following query:

> Query [Customers]Lastname is equal to Smith Except
> [Customers]Firstname is equal to Glenn.

The “Except” operator tells DataStrike! to find those records where the first condition is met except when the second conditions is met.

Lastname	Firstname
Steve	Smith

Set Operators

Set operators allow the 4D User to specify which group of records from the current table the query will be performed on.

Figure 30: Set Operators

The first Set operator, “Query All Records”, is chosen as default for each new query: The table shown on the following page reviews all possible Set Operator Icons, the purpose and a brief description.

Set Operator Icons

Icon	Set Operator	Description
	Query all records	Performs the query against the entire table.
	Query in Selection	Performs the query only against the current selection.
	Add results to current collection	Performs the query and adds the resulting records to the current selection.
	Remove results from Current Collection	Performs the query and removes the resulting records from the current selection.

Executing Queries and Query Filters

To execute a Query from the Query Editor, click the Query button once the query is built to your specifications.

Figure 31: The Query Button

To execute a Query from the Query Filters Dialog, click the Query button when the desired Query Filter is selected (highlighted).

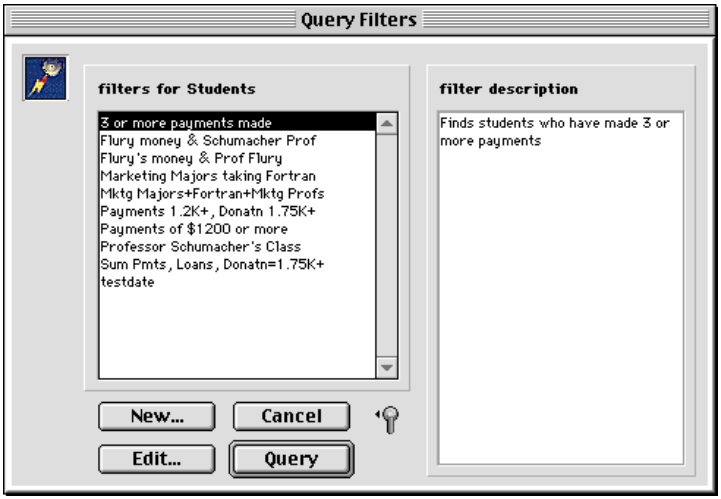


Figure 32: The Query Filters Dialog

Whenever a User attempts to query a table which has more than one path to the destination table, the query becomes confused until the User resolves a circular path. Whenever DataStrike! encounters this situation, the Circular Path dialog box appears.

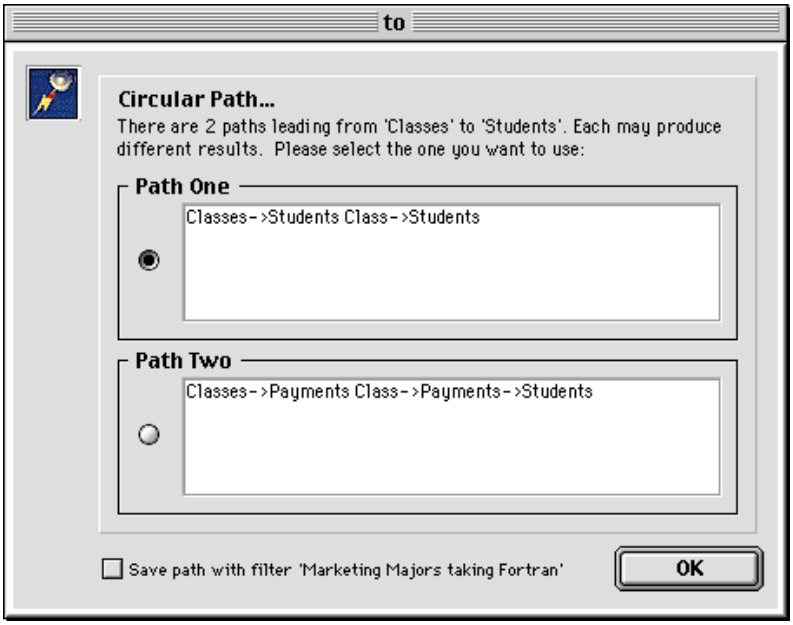

to
Circular Path...
There are 2 paths leading from 'Classes' to 'Students'. Each may produce different results. Please select the one you want to use:
Path One
Classes->Students Class->Students
Path Two
Classes->Payments Class->Payments->Students
Save path with filter 'Marketing Majors taking Fortran'
OK

Figure 33: The Circular Path Dialog

If a “Circular Path” Query is initiated from a “Saved” Filter, the User can save the path by clicking the “Save Path” checkbox (below) and make it a permanent part of the Query Filter.

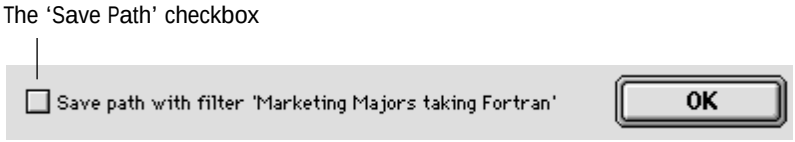


Figure 34: Electing to save a Specific Path within a Query Filter

If a User changes their mind on the path of an existing Query and elects to change the path, they can simply hold the “option” key down and the Circular Path dialog reappears.

Circular Relationships

In a 4th Dimension database, “linked” lines are drawn between one or more tables to specify a relationship between two tables. These “relations” allow the 4D developer to take advantage of 4D included forms, simplified cross-table searching and other various tools within 4D’s various Editors. When table relations become complicated, the 4D developer must tackle various issues such as the correct managing of selections within both the current and related tables. When one 4D table has two or more paths to another 4D table, the table relationship is referred to as a circular path.

DataStrike! will handle up to two simple (non-adjacent) circular paths that don't involve subtables. In certain instances, circular paths are justifiable in the design of a database design. **In version 1.1, DataStrike! supports up to 2 circular (or multiple) paths for any table**.

This section is intended to be a primer for 4D developers unfamiliar with circular table relationships as well as a guide for handling potential instances where more than 2 circular relationships exist. In certain data relationships, 4th Dimension cannot query properly when using manual relationships. Because of this, DataStrike utilizes Automatic relationships in lieu of Manual relationships.

DataStrike turns on Automatic Relations (where manual relations exist) prior to entering the DataStrike! Query Editor and turns them off upon exiting. See the QEr_RunQuery method.

Definitions:

Path: a set of relational links that joins one table to another.

Circular path: a path, which forms a circle or loop.

Simple circular path: a circular path where there are only two paths between any two tables forming the circle.

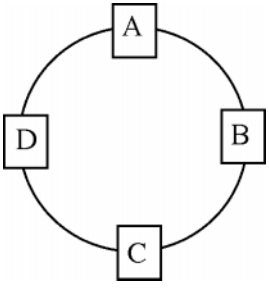


Figure 35: 2 Circular Paths

In Figure 35, there are two (2) paths between table B and table D. DataStrike! supports this data relationship in version 1.1.

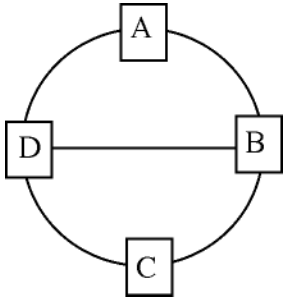


Figure 36: 3 Circular Paths

In Figure 36, there are three (3) paths between table B and table D. DataStrike! does not support this data relationship in version 1.1.

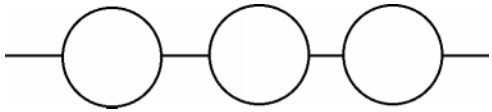


Figure 37: Multiple Non-adjacent Circular paths

In Figure 37, there are several non-adjacent circular paths. DataStrike! supports this data relationship in version 1.1.

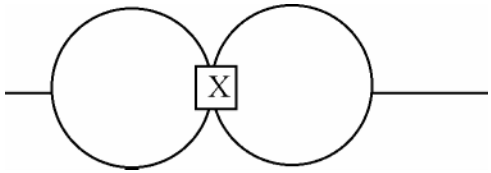


Figure 38: Multiple Adjacent Circular paths

The multiple circular paths are adjacent (table X belongs to both circular paths). DataStrike! does not support this data relationship in version 1.1.

Circular Relationships

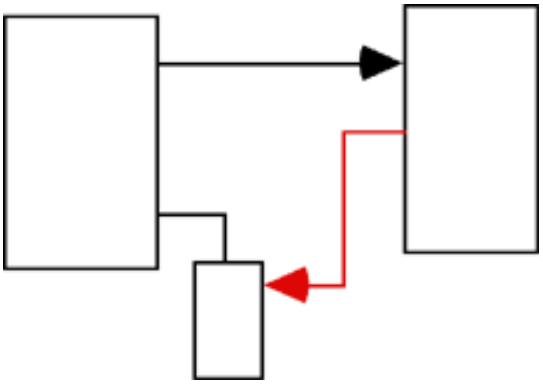


Figure 39: Circular paths with Subtables

This path contains a subtable. DataStrike! does not support this data relationship in version 1.1.

Solving Complex Circular Path Problems

Beginning with version 1.1, DataStrike! provides a developer dialog which serves as a tool for describing the specific data paths the DataStrike! Search Editor will use when searching. The following dialog appears only when three or more paths are detected between two tables. Because DataStrike! only supports two (or fewer) paths between two tables, it will be necessary to tell DataStrike! about any paths which should be ignored (or severed). Below is the dialog which will appear when DataStrike! executes the “QE UPDATE DATA” method and encounters three or more paths (a complex path) between two tables.

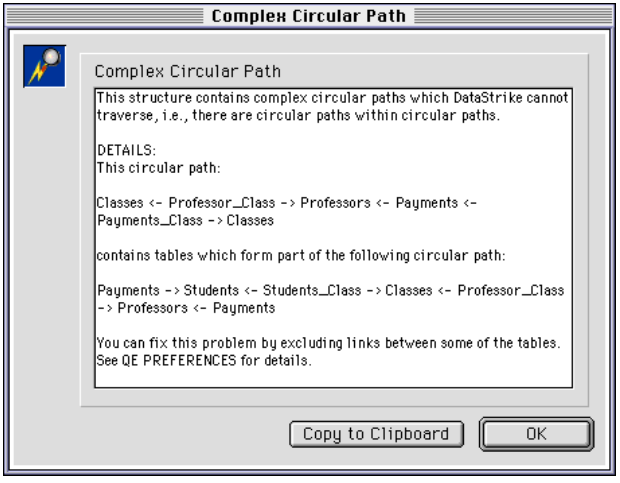


Figure 40: Complex Circular Paths (which are not allowed)

When this dialog appears, it means the QE UPDATE DATA method did not complete successfully - and will need to be rerun. Before rerunning QE UPDATE DATA, it is recommended (and required) that the 4D Developer sever those relational links which are not needed for searching. As a developer, you will probably want to copy the descriptive information contents to the clipboard by clicking on the “Copy to ClipBoard” button. This is important so you can determine how to proceed. DataStrike! cannot make this decision for you because it requires an understanding of how your database is to be used.

Once you determine which path(s) are unwanted, you can call the DataStrike! project method “QEs_ExcludeLinks” to sever the unwanted link(s). To sever a relational link without altering the 4D structure, QEs_ExcludeLinks requires one parameter which is either a pointer to a field name or a pointer to a table name.

The purpose of the QEs_ExcludeLinks is to designate those tables/fields which will not be used in construction of relational paths.

Because DataStrike! will only display an unsupported complex path one at a time, it will be necessary to rerun the QE UPDATE DATA method. If the QE UPDATE DATA method encounters an additional complex path, the process just described will occur again.

When passing a field name pointer, the field must be a related many link from the “Many” table or a pointer to a table.

- Severing a “Many” table link

Example: QEs_ExcludeLinks (->[Table]ManyField)

This Many Table link will be the database field which points directly to the One table key. By passing this type of field, the link between the two tables will be avoided and the Search Editor will be unaware of the path relationship existing in the 4D structure.

- Severing a Table

Example: QEs_ExcludeLinks (->[Table])

Passing a pointer to a table when calling QEs_ExcludeLinks will sever all links to that table.

Another possible way to avoid problems within DataStrike! where 3 or more circular paths exist would be to:

A) Temporarily remove links that create the “won’t work” situations.
B) Run QE UPDATE DATA to build DataStrike mapping records in QE Data table.
C) Replace links removed in step A.

Alternatively, if possible, 4D Developers can remove links that create any of the above “won't work” situations. Candidates would be links that are seldom used.

Symbols

◊QEkl_EditorForm 7
◊QEkl_MethodStrings 7
◊QEkl_OtherForms 7

Numerics

4D Insider 2, 3

A

Add Line Button 22
And Conjunction 23
Average 19

C

Choice Lists 17
choice lists 6
Circular Path 26, 27
Clear All Button 21
Count 20
Create Group 21

D

Delete Line Button 22

E

Except 24
Except Conjunction 24

F

Field Pop-up List 17
Field Selection 16
Fields, Comparison Operators and Values 16
Figure 13
    DataStrike! Query Filters Dialog 15
Figure 14
    Table Pulldown Menu 15
Figure 15
    Field Selection 16
Figure 16
    Field Pop-Up List 17
Function Dialog 19

I

Insert Line Button 22
Insider Library 2
Installation 2
Introducing DataStrike!™ 1

M

Mandatory Developer Steps 3
Manual or Automatic Relations 4
Mapping records 5
Max 19
Min 19

O

Optional Developer Steps 3, 5
Or Conjunction 23

Q

QE ON STARTUP 4
QE PREFERENCES 7
QE UPDATE DATA 5
QE_bCallQE 4
QE_Register 4
Query Button 25
Query button 25
Query Editor 14, 25
Query Filter record 5
Query Filters Dialog 15, 25
Query Groups 23

R

ResEdit 7

S

Show Format 12
Show Resources 12
Sum 19

T

Table Pulldown Menu 15

V

version record 5